Statistics: Covariance

Covariance

- Measures linear relationship between two variables

Statistics: Covariance

- Covariance (Population)

$$\mathrm{Cov}\,(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n}$$

Where

Σ = the sum of

Xi = individual datum value

$\bar{X}$ = population mean

n = the number of datum in the population

Statistics: Covariance

- Covariance (Sample)

$$\mathrm{Cov}\,(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n - 1}$$

Where

Σ = the sum of

Xi = individual datum value

$\bar{X}$ = sample mean

n = the number of datum in the sample

Statistics: Covariance

$$\mathrm{Cov}\,(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n - 1}$$

X	Y	$\bar{X}$	$\bar{y}$	(X-$\bar{X}$)	(Y-$\bar{Y}$)	(X-$\bar{X}$)(Y-$\bar{Y}$)
Temerature	Customer					
100	60	87.5	180	12.5	-120	-1495.83
95	98	87.5	180	7.5	-81.7	-612.5
90	100	87.5	180	2.5	-79.7	-199.167
85	200	87.5	180	-2.5	20.3	-50.8333
80	300	87.5	180	-7.5	120	-902.5
75	320	87.5	180	-12.5	140	-1754.17

Σ (X-$\bar{X}$)(Y-$\bar{Y}$) = -5015

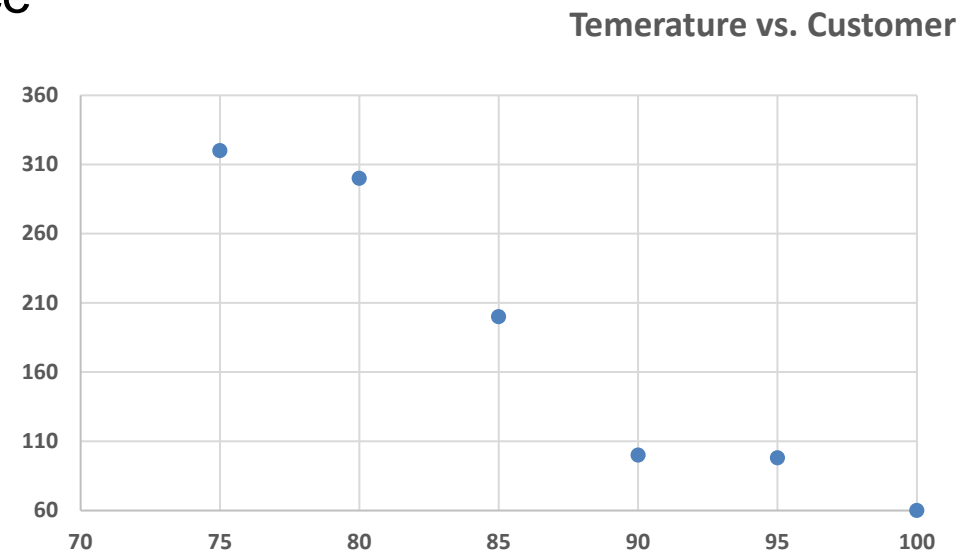
Σ (X-$\bar{X}$)(Y-$\bar{Y}$) / n-1 = -5015 / 5

Cov(X,Y) = -1003

Statistics: Covariance

X	Y
Temerature	**Customer**
100	60
95	98
90	100
85	200
80	300
75	320

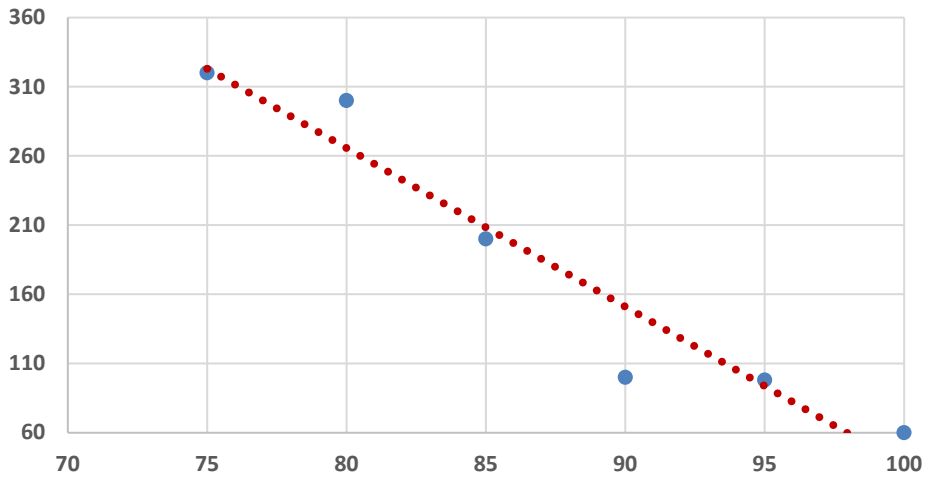
Cov(X,Y) = -1003



Statistics: Covariance

X	Y
Temerature	**Customer**
100	60
95	98
90	100
85	200
80	300
75	320

Cov(X,Y) = -1003



Statistics: Covariance

X	Y	$\bar{X}$	$\bar{y}$	$(X-\bar{X})$	$(y-\bar{y})$	$(X-\bar{X})(y-\bar{y})$
Chips	**Weight**					
2	120	4.83	185	-2.83	-65.3	185.111
5	207	4.83	185	0.17	21.7	3.61111
3	122	4.83	185	-1.83	-63.3	116.111
8	247	4.83	185	3.17	61.7	195.278
5	227	4.83	185	0.17	41.7	6.94444
6	189	4.83	185	1.17	3.67	4.27778

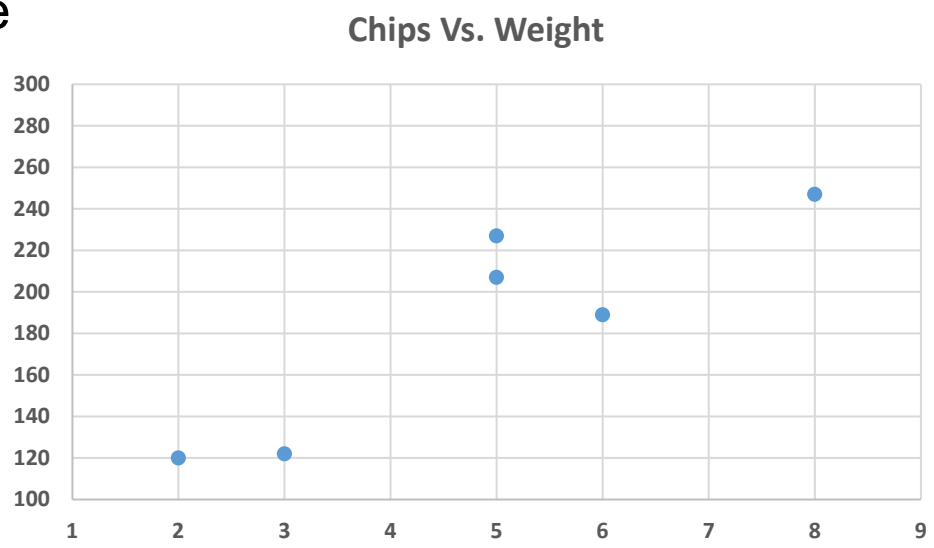
$\Sigma (X-\bar{X})(Y-\bar{Y}) = 511.33$

$\Sigma (X-\bar{X})(Y-\bar{Y}) / n-1 = 511.33 / 5$

Cov(X,Y) = 102.26

Statistics: Covariance

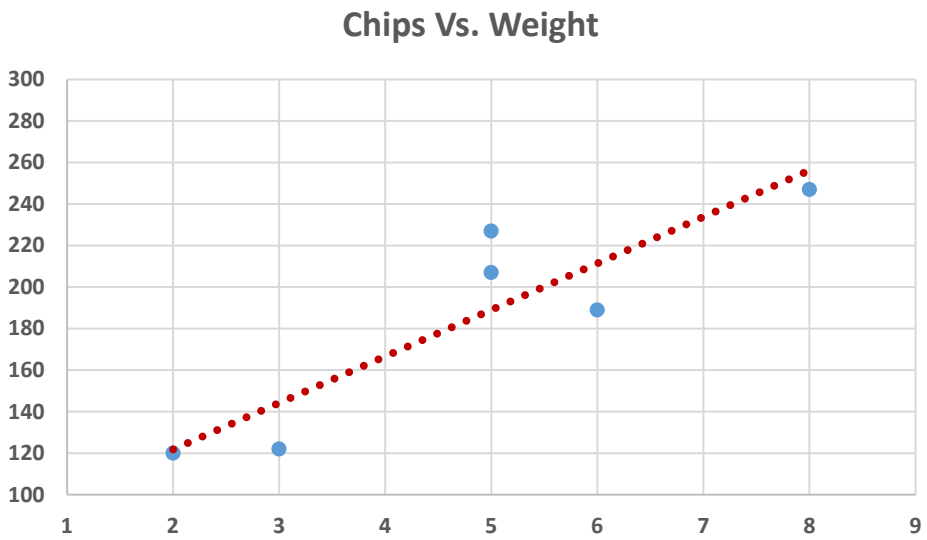
X	Y
Chips	Weight
2	120
5	207
3	122
8	247
5	227
6	189



Cov(X,Y) = 102.26

Statistics: Covariance

X	Y
Chips	Weight
2	120
5	207
3	122
8	247
5	227
6	189



Cov(X,Y) = 102.26

Statistics: : Covariance

- **Cov(X,Y) = 0 ?**

Statistics: : Covariance

- + means positive linear relationship
- - means negative linear relationship
- 0 means no linear relationship

- It only gives the direction of the relationship
- No indication of the strength of the relationship

Statistics: Correlation

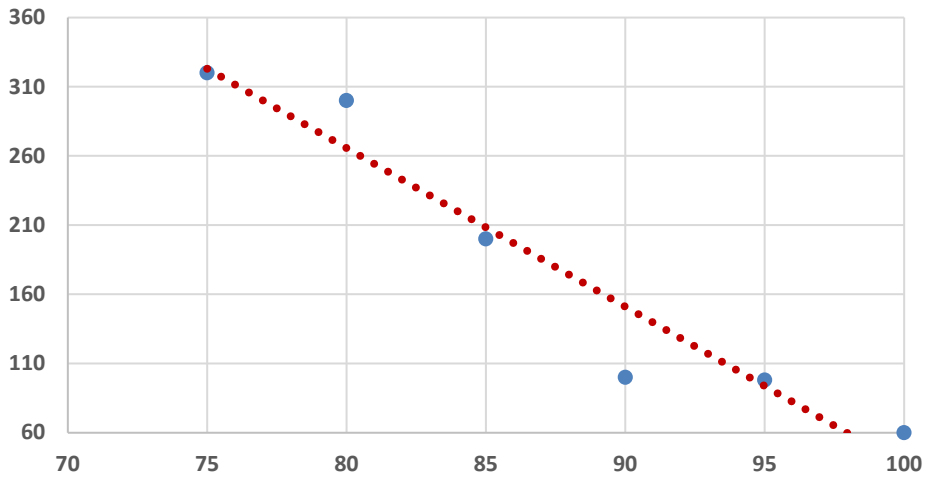
Correlation

- Measures relationship and strength of the relationship between two variables

Statistics: Covariance

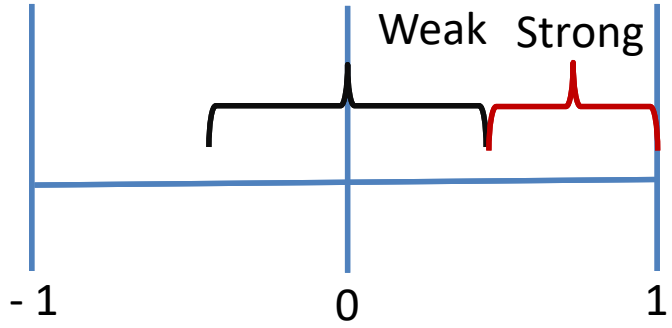
X	Y
Temerature	**Customer**
100	60
95	98
90	100
85	200
80	300
75	320

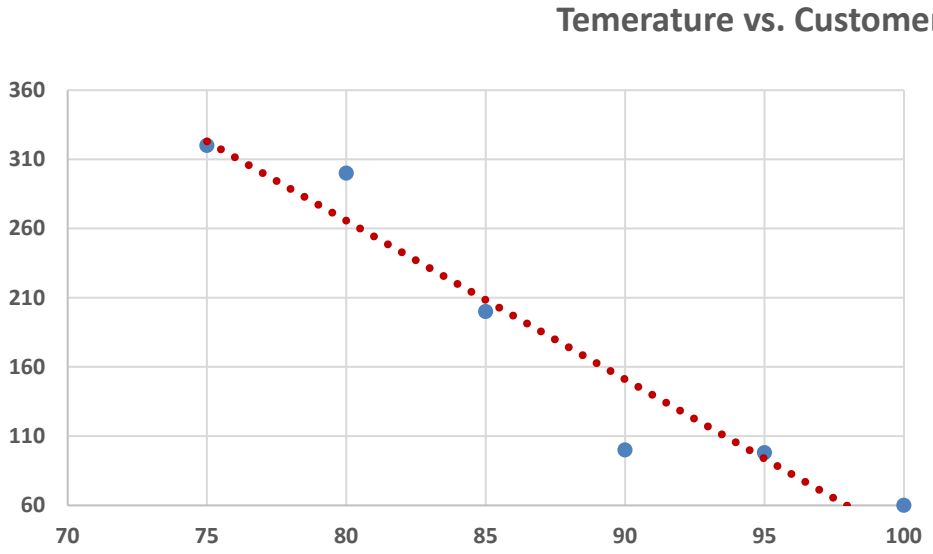
Cov(X,Y) = -1003



Statistics: Correlation

- Pearson Correlation Coefficient
  - Pearson r

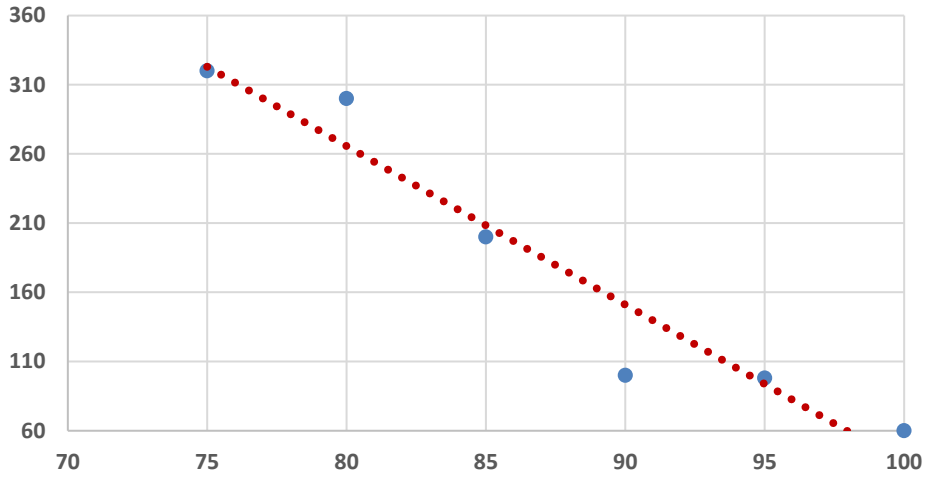




$$r = Cov(X,Y) / s_x s_y$$

Statistics: Correlation

X	Y
Temerature	Customer
100	60
95	98
90	100
85	200
80	300
75	320



Cov(X,Y) = -1003

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

$S_x = 9.35$

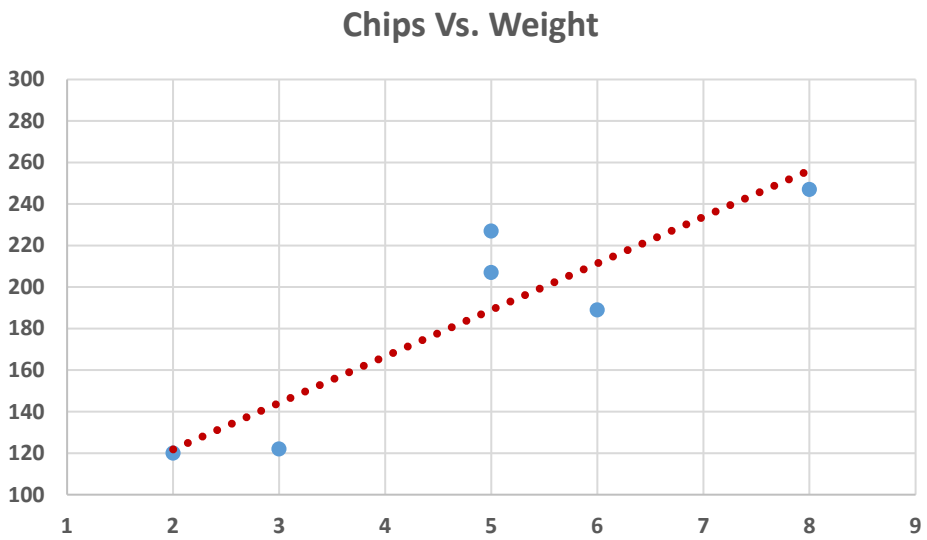
$S_y = 111.29$

$r = Cov(X,Y) / s_x s_y = -1003/(9.35 * 111.29)$

$r = -0.9635$

Statistics: Correlation

X	Y
Chips	Weight
2	120
5	207
3	122
8	247
5	227
6	189



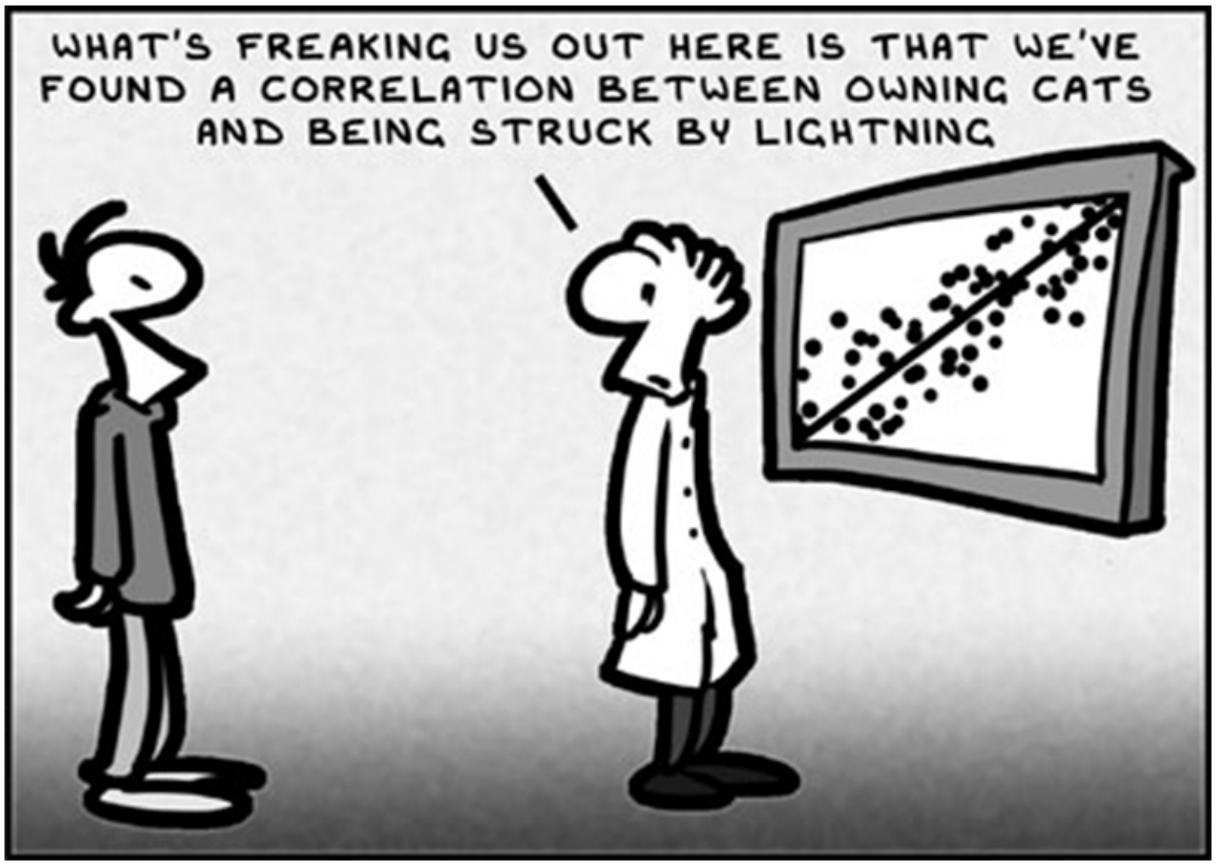
Cov(X,Y) = 102.26

$r = Cov(X,Y) / s_x s_y = 0.8948$

Statistics: Correlation

- Correlation does not equal to causation

Statistics: Correlation



Python Introduction and Installation

A programming language created by Guido van Rossum and was made available in 1991.

Python can be used for:

Web programs,
Scripts,
Math,
Big data management,
Data analytics

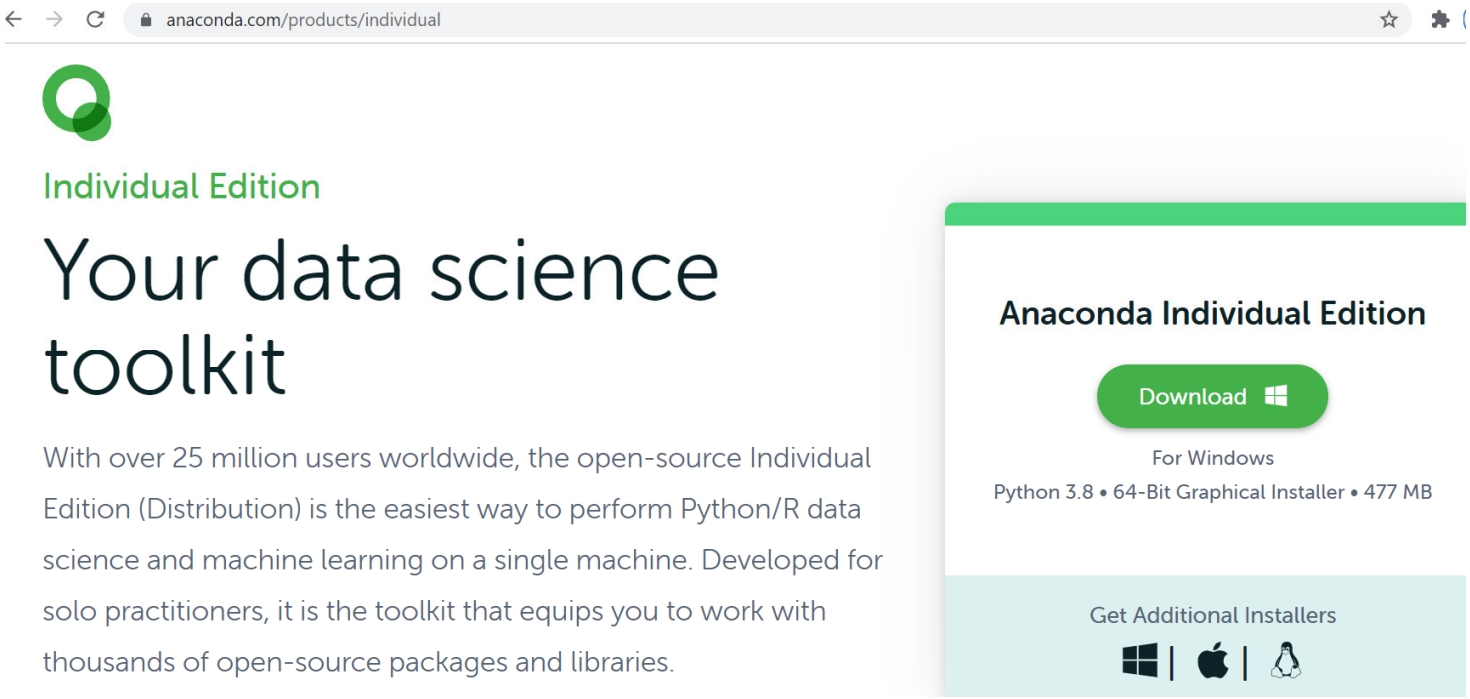
Python Introduction and Installation

- Why Python?
- Portable
  - Windows, Mac, Linux, Raspberry Pi, etc.
- “common sense” syntax
  - Easy to learn
  - Simplicity
- Interpretive programming language
- Highly forgiven
  - procedural, functional, object-orientated

Python Installation

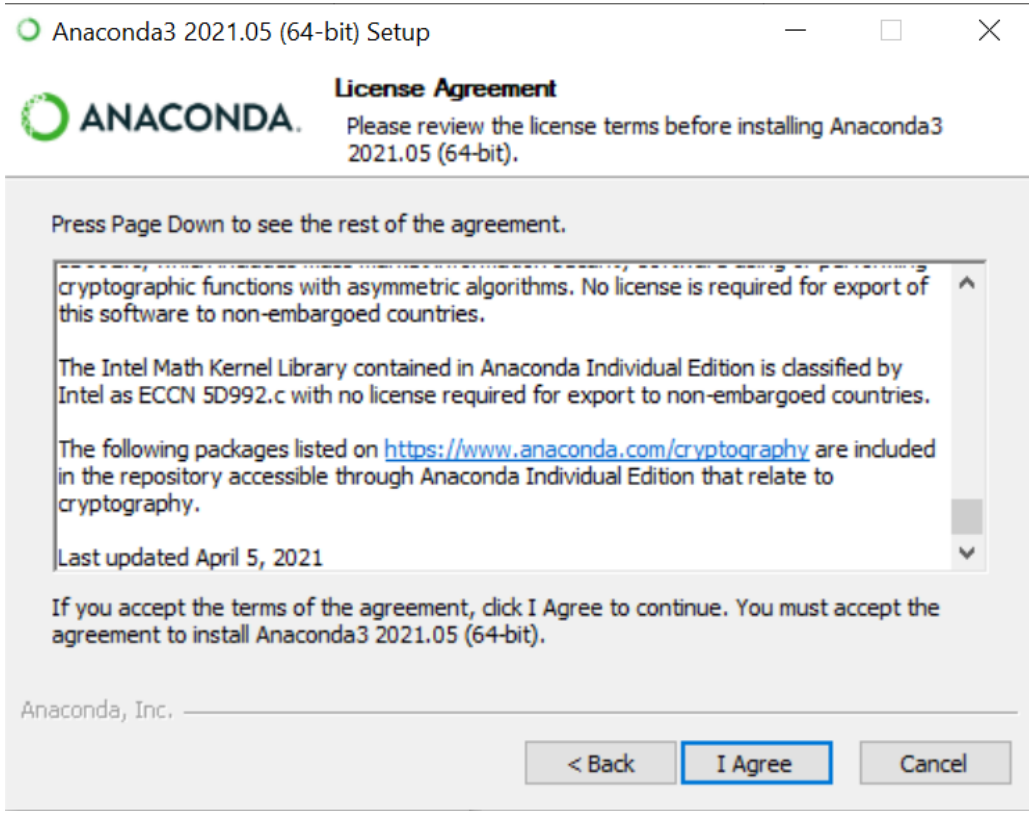
- Anaconda
  - Python distribution and related libraries

Python Installation Go to Anaconda.com/download

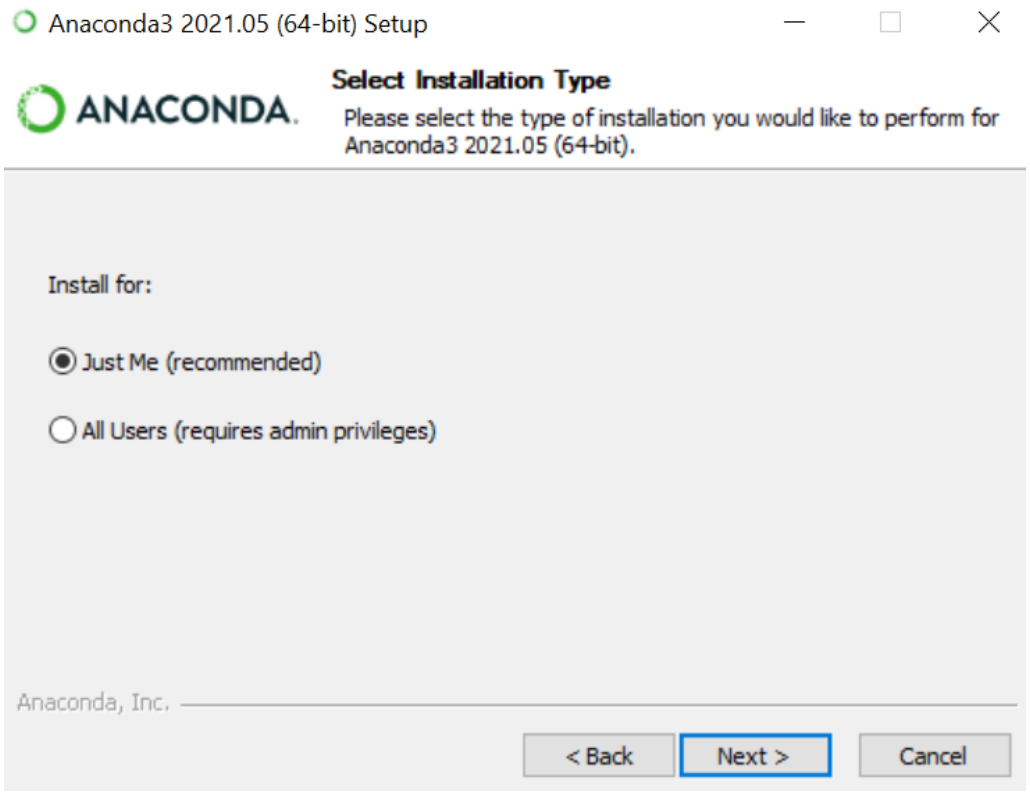


Python Installation Go to Anaconda.com/download

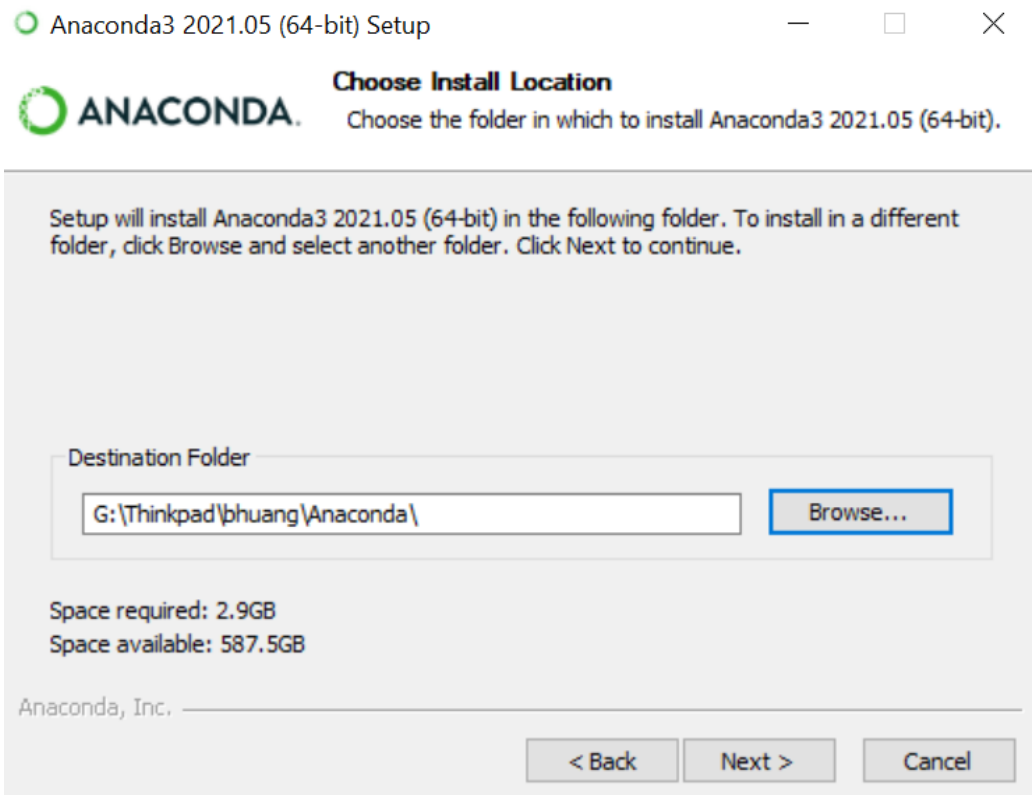
Python Installation



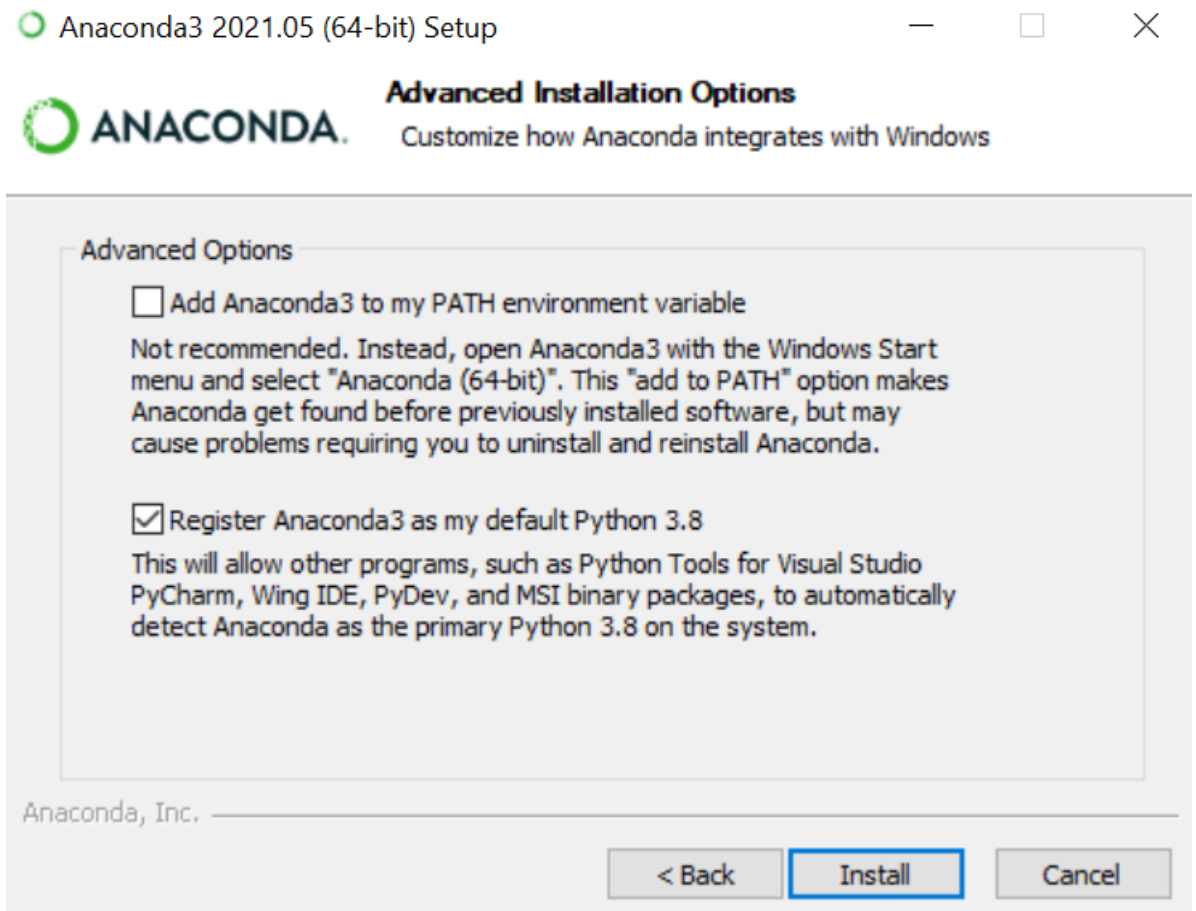
Python Installation



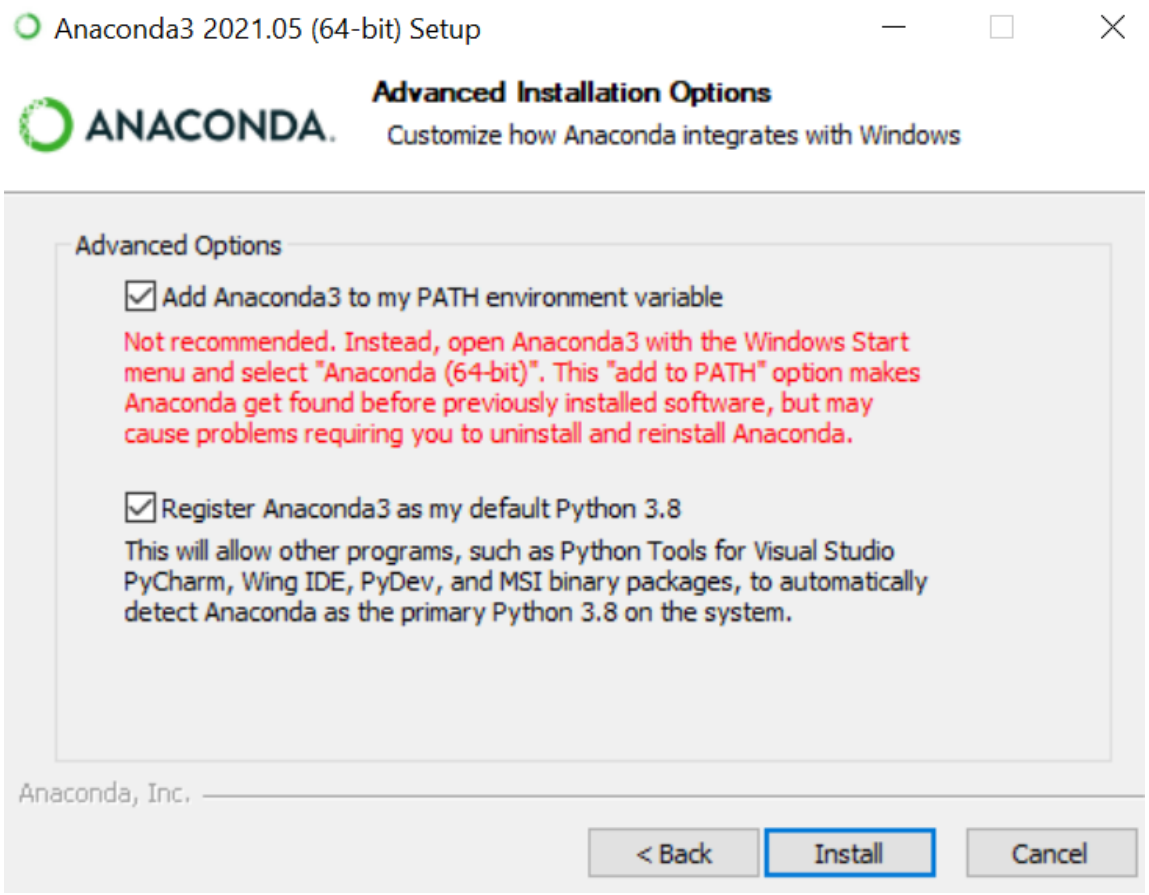
Python Installation



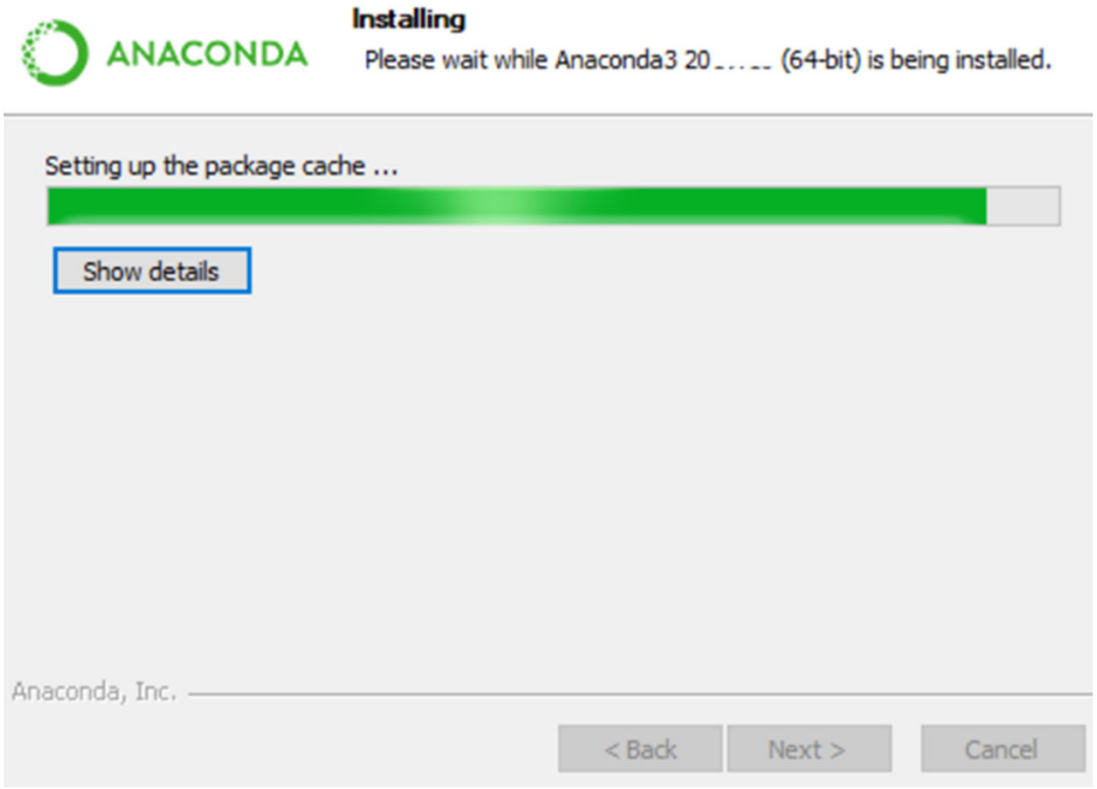
Python Installation



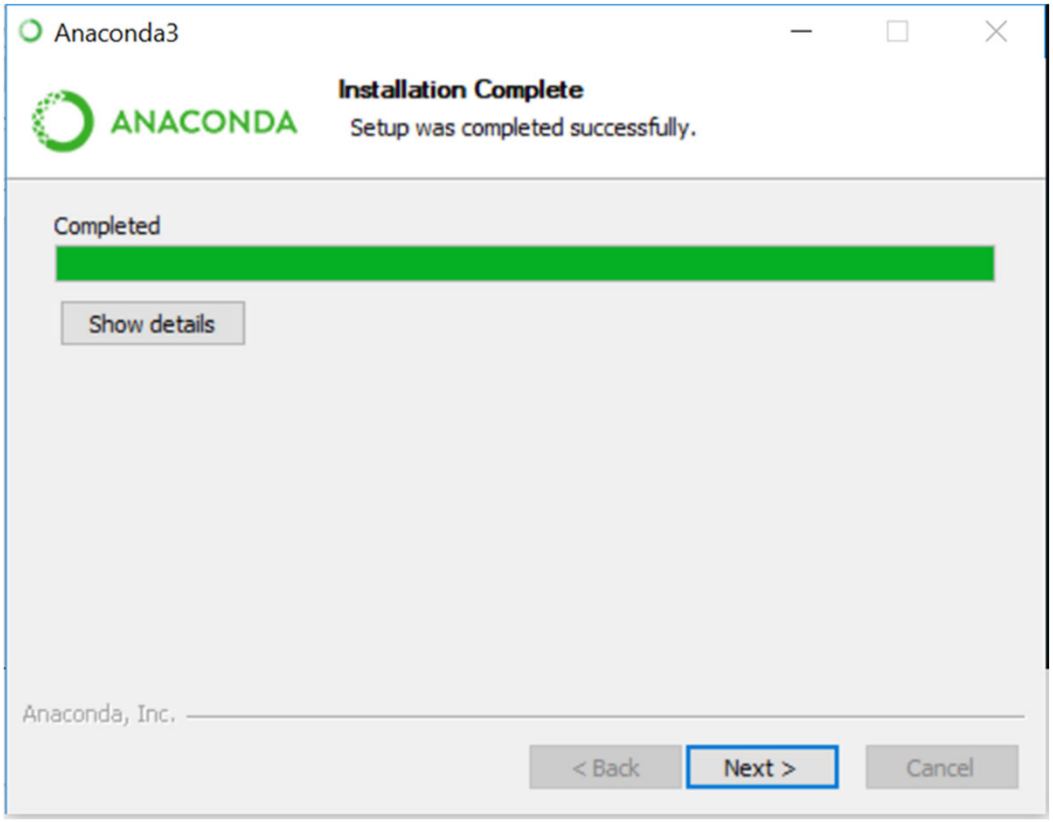
Python Installation



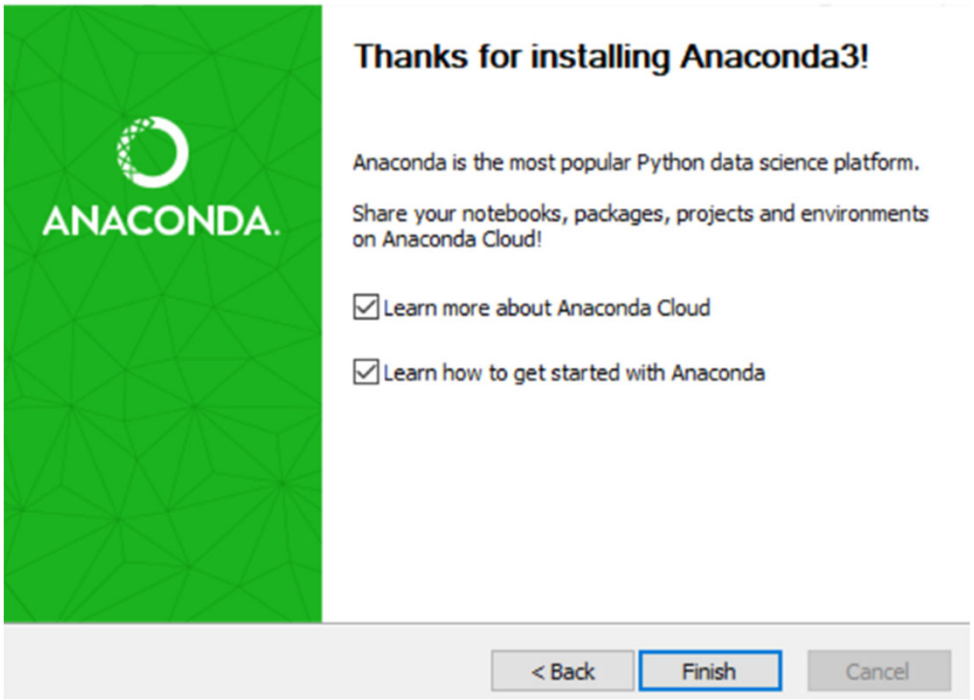
Python Installation



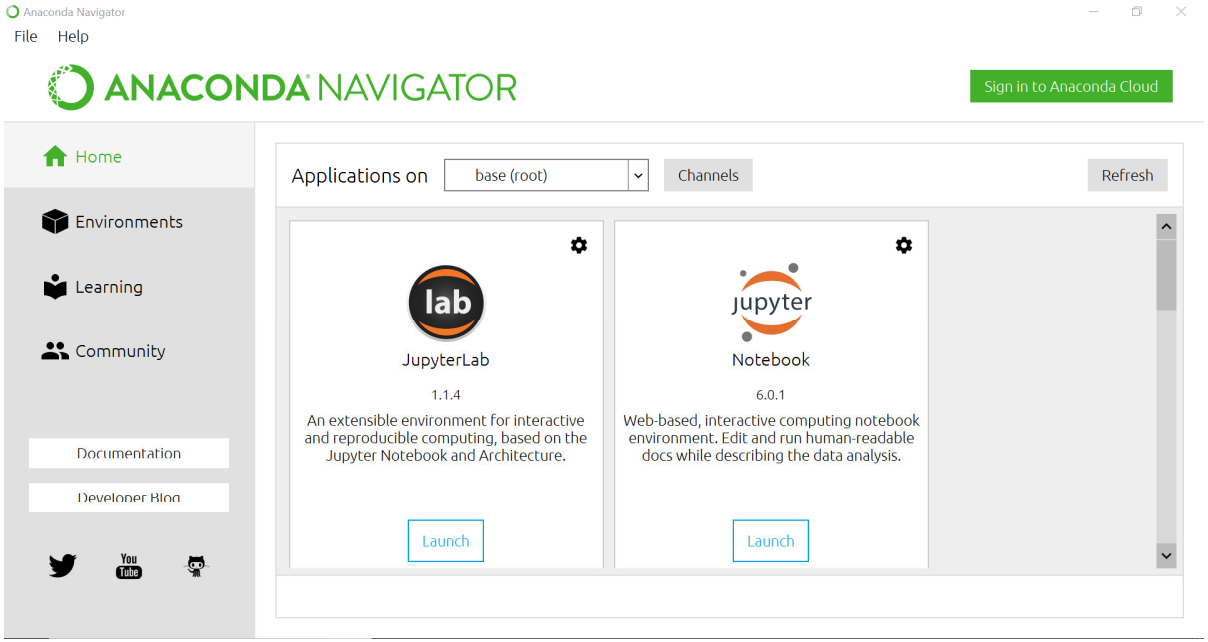
Data Science: Python Installation



Python Installation



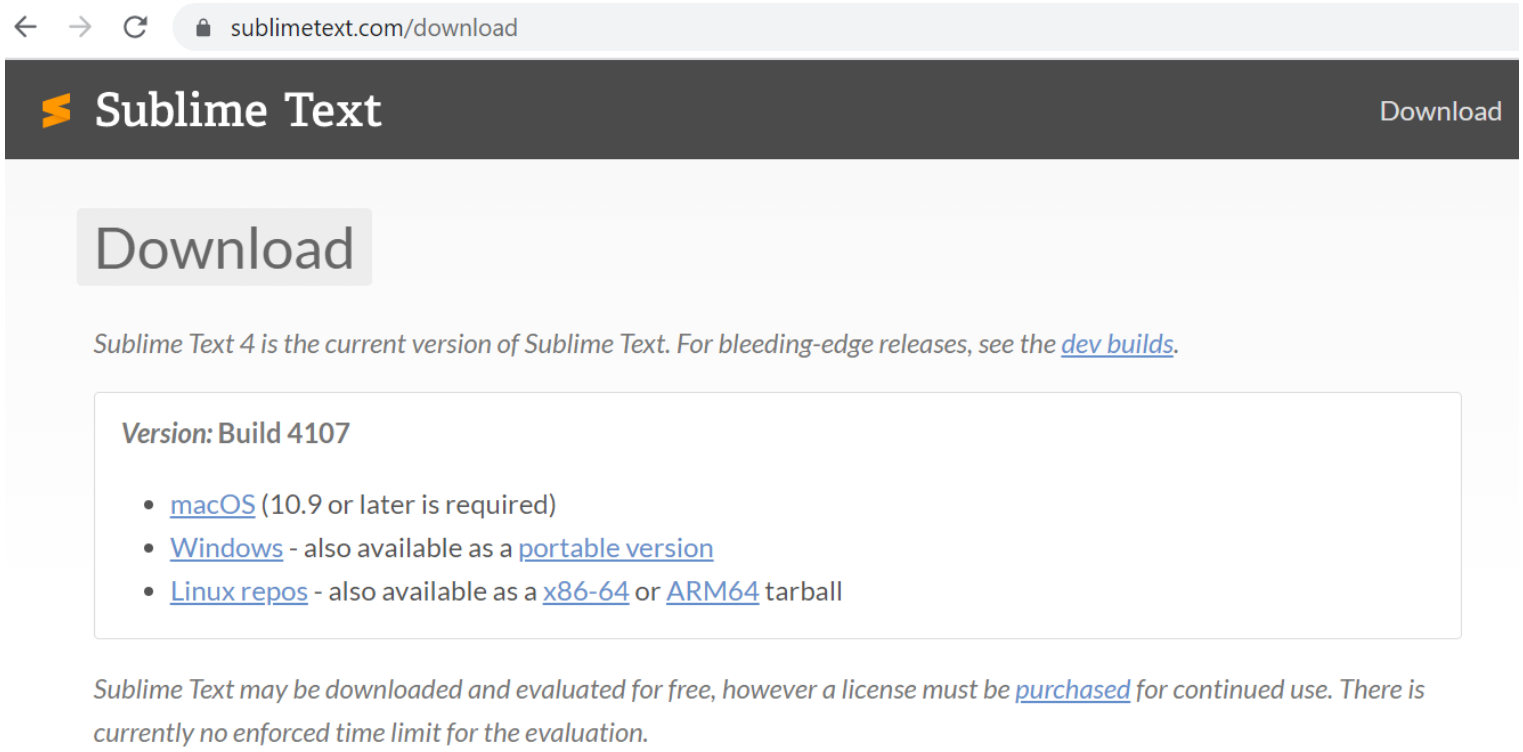
Python Installation



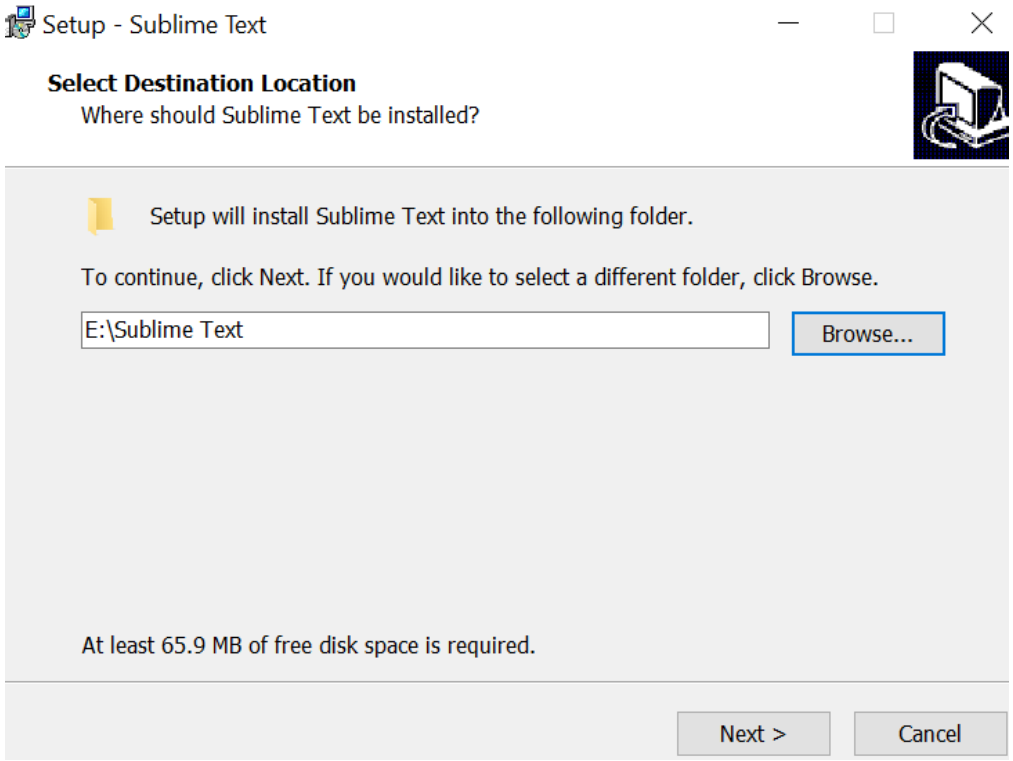
Python Introduction and Installation

Python Sublime Text Editor

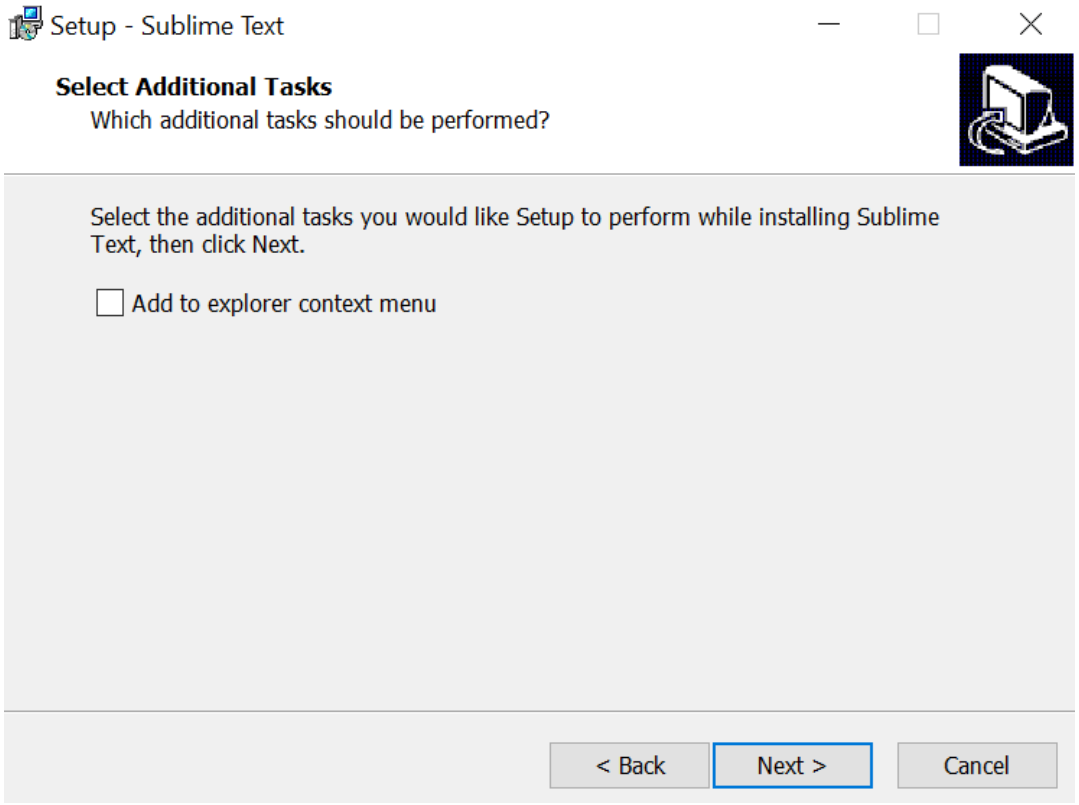
Python Sublime Text Editor • Go to https://www.sublimetext.com/download



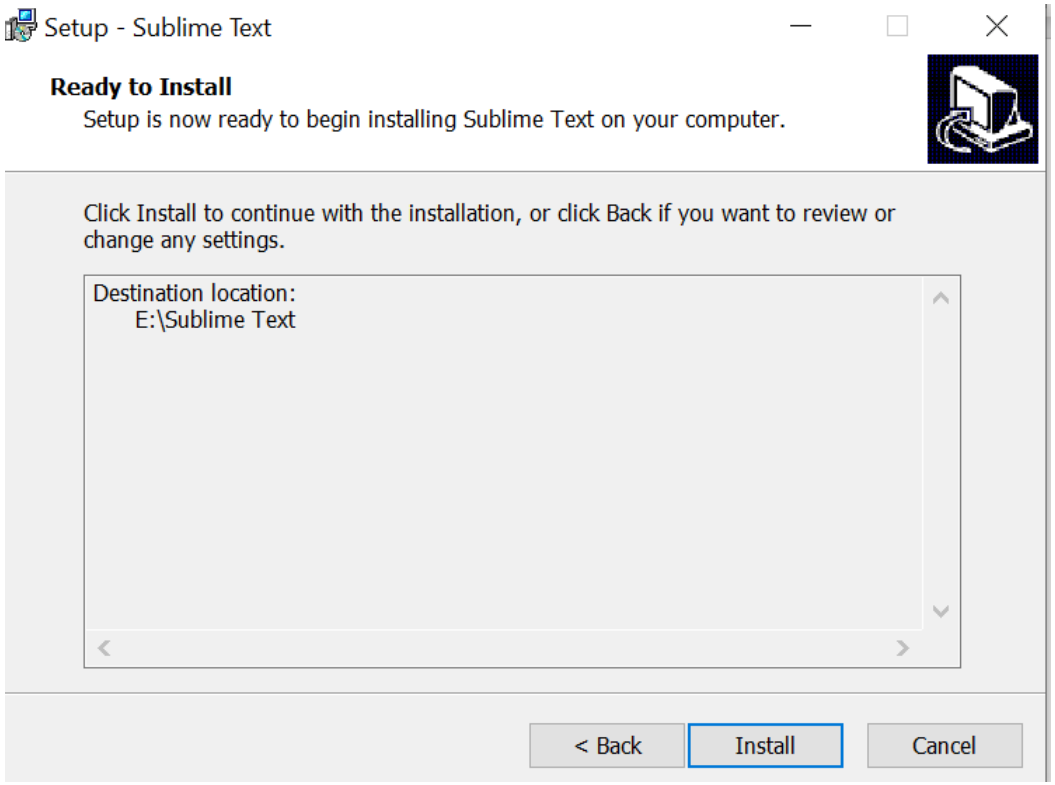
Python Sublime Text Editor



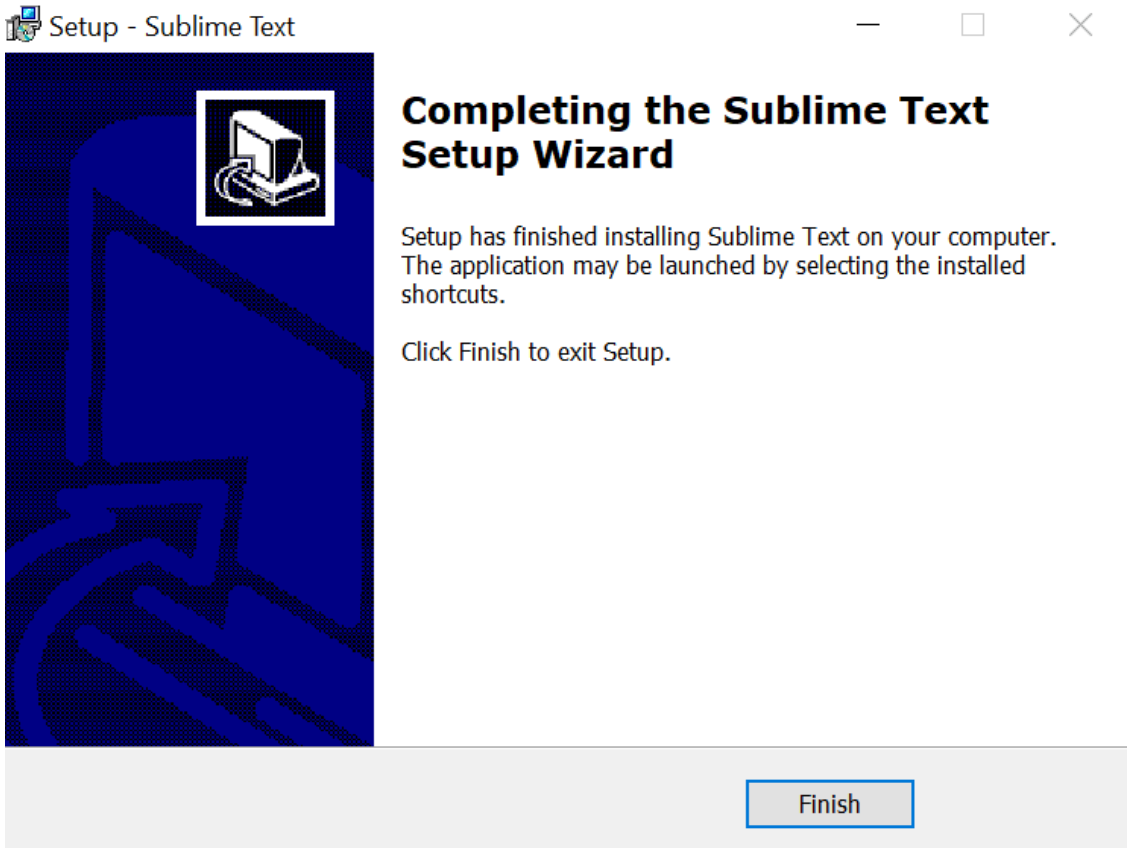
Python Sublime Text Editor



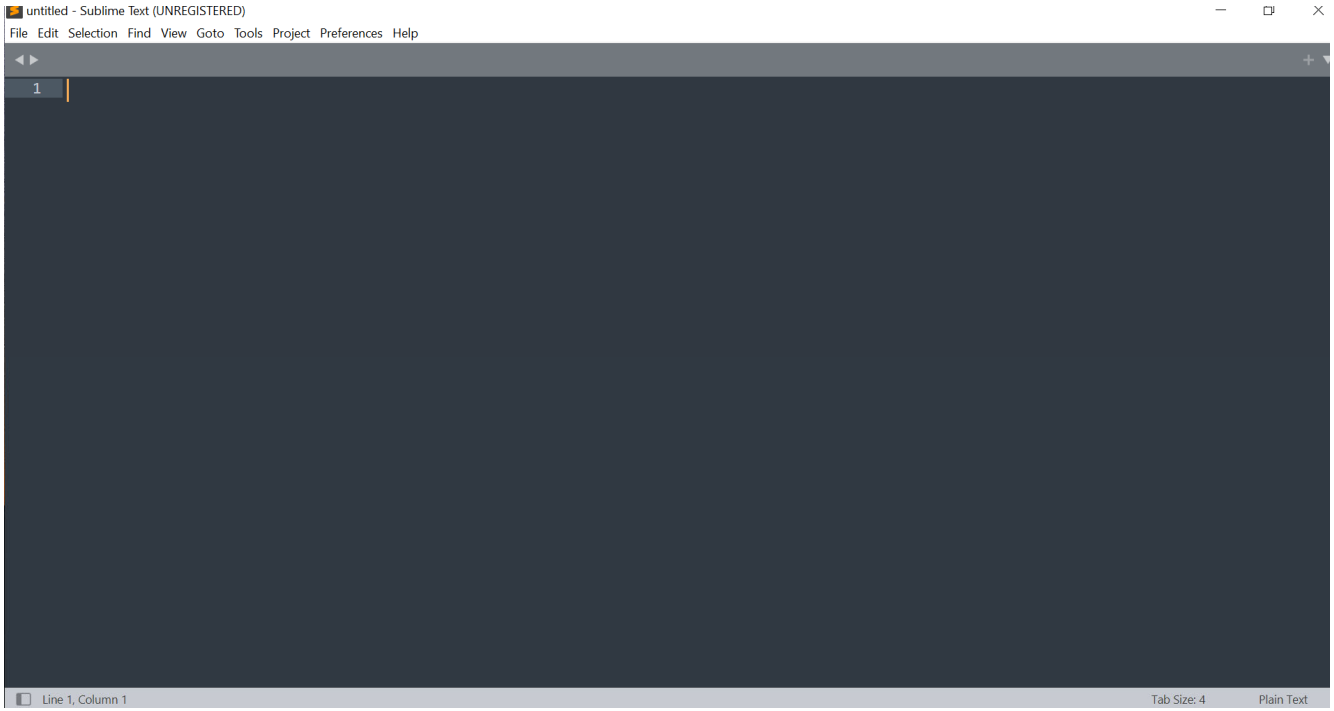
Python Sublime Text Editor



Python Sublime Text Editor



Python Sublime Text Editor



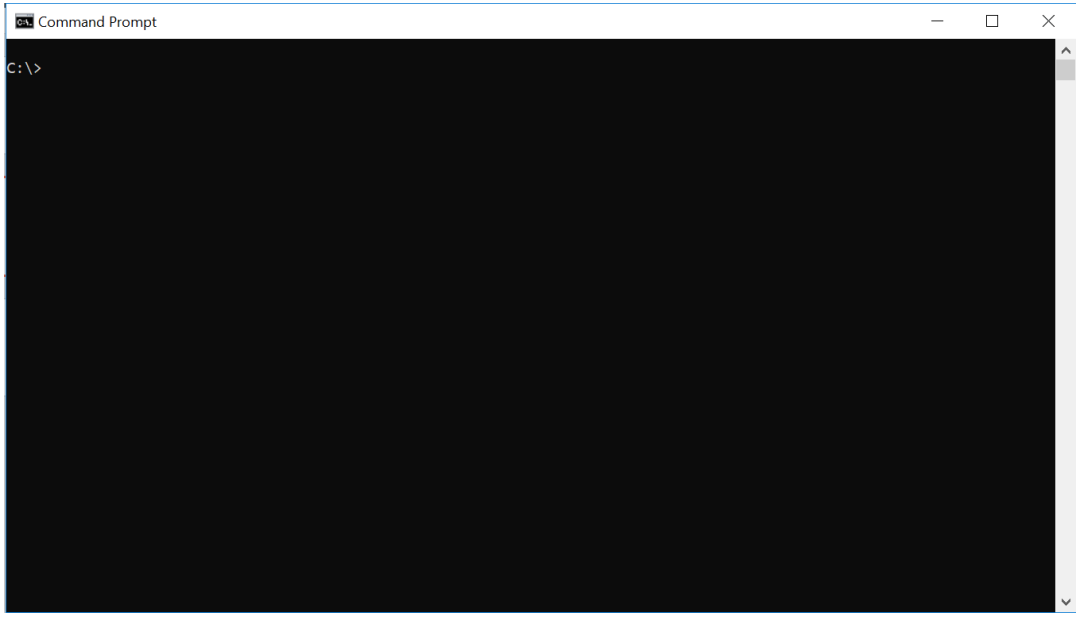
Data Science: Python Sublime Text Editor

Data Science: Python Prompt via Command Prompt

Python Prompt via Command Prompt

To get to the Python Prompt:

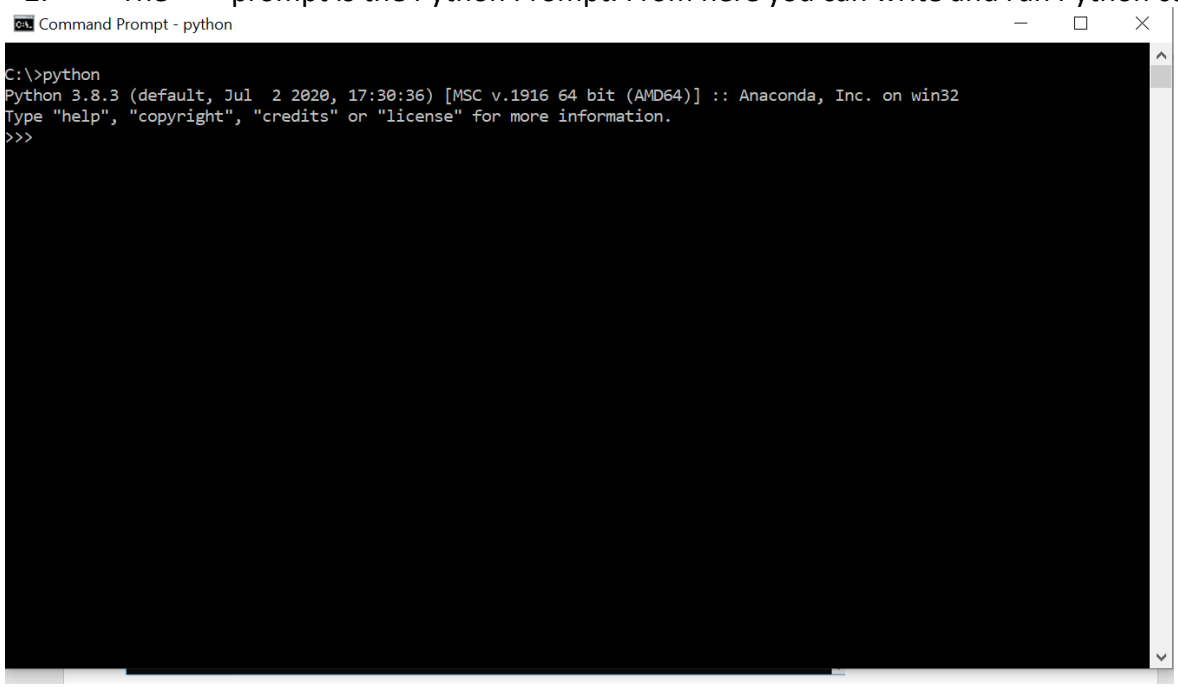
1. Launch Command Prompt (The Mac OS equivalent is Terminal)
    1. Go to bottom left corner of Windows where you can type
        1. Enter CMD then select Command Prompt from the list



Python Prompt via Command Prompt

To get to the Python Prompt from Command Prompt:

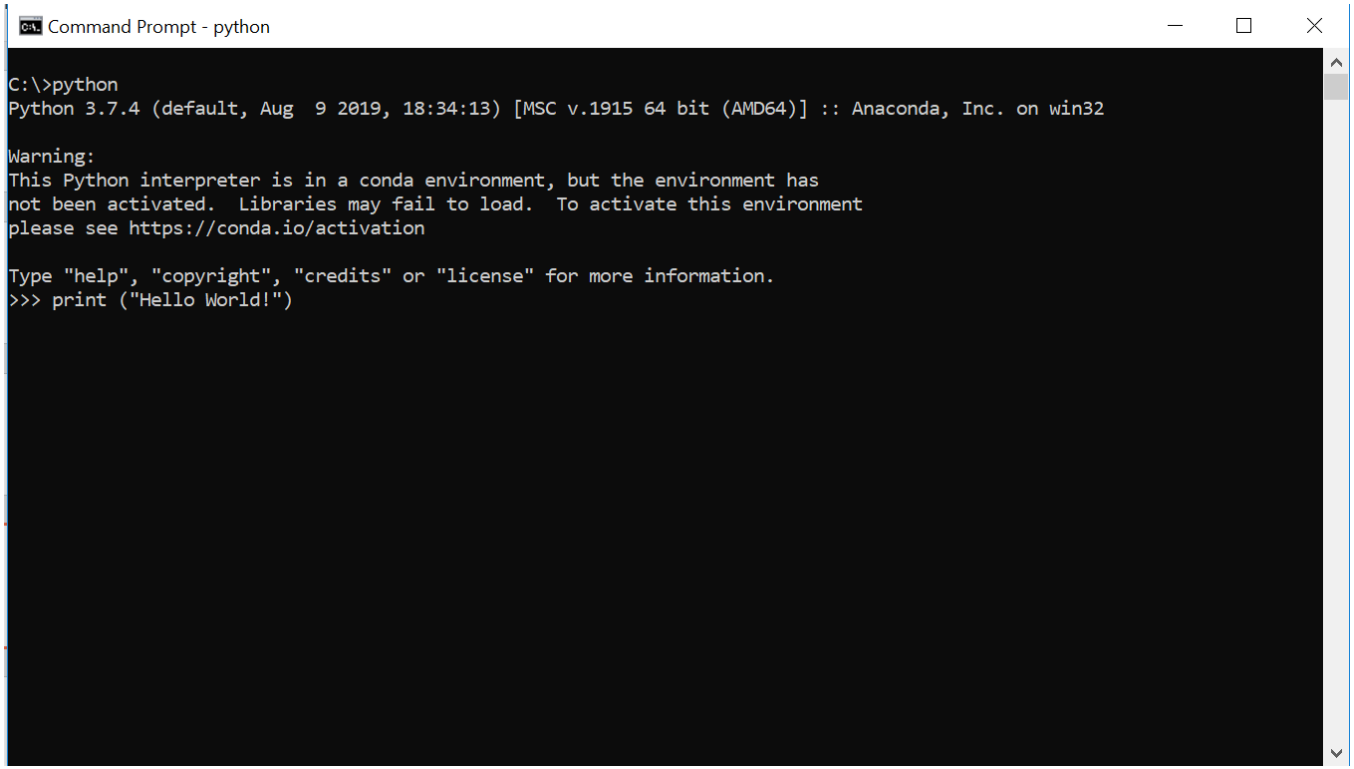
1. Type python at the prompt then hit ENTER
2. The >>> prompt is the Python Prompt. From here you can write and run Python code



Python Prompt via Command Prompt

Let’s write our first Python program via Python prompt

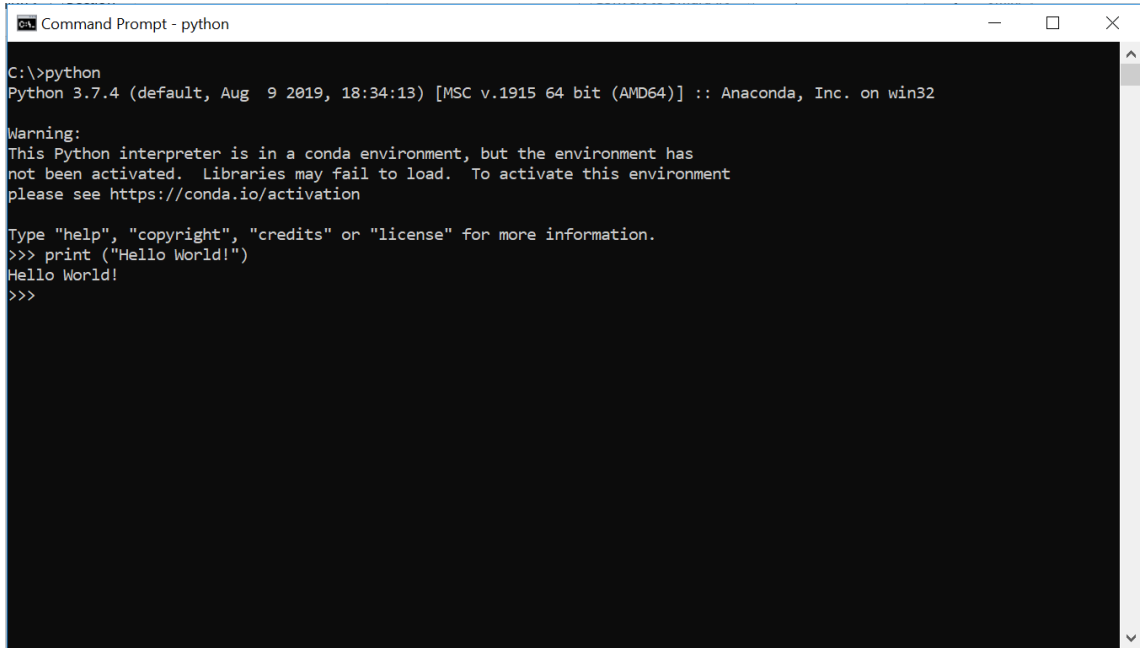
1. print (“Hello World!”)



Python Prompt via Command Prompt

Let’s write our first Python program via Python prompt

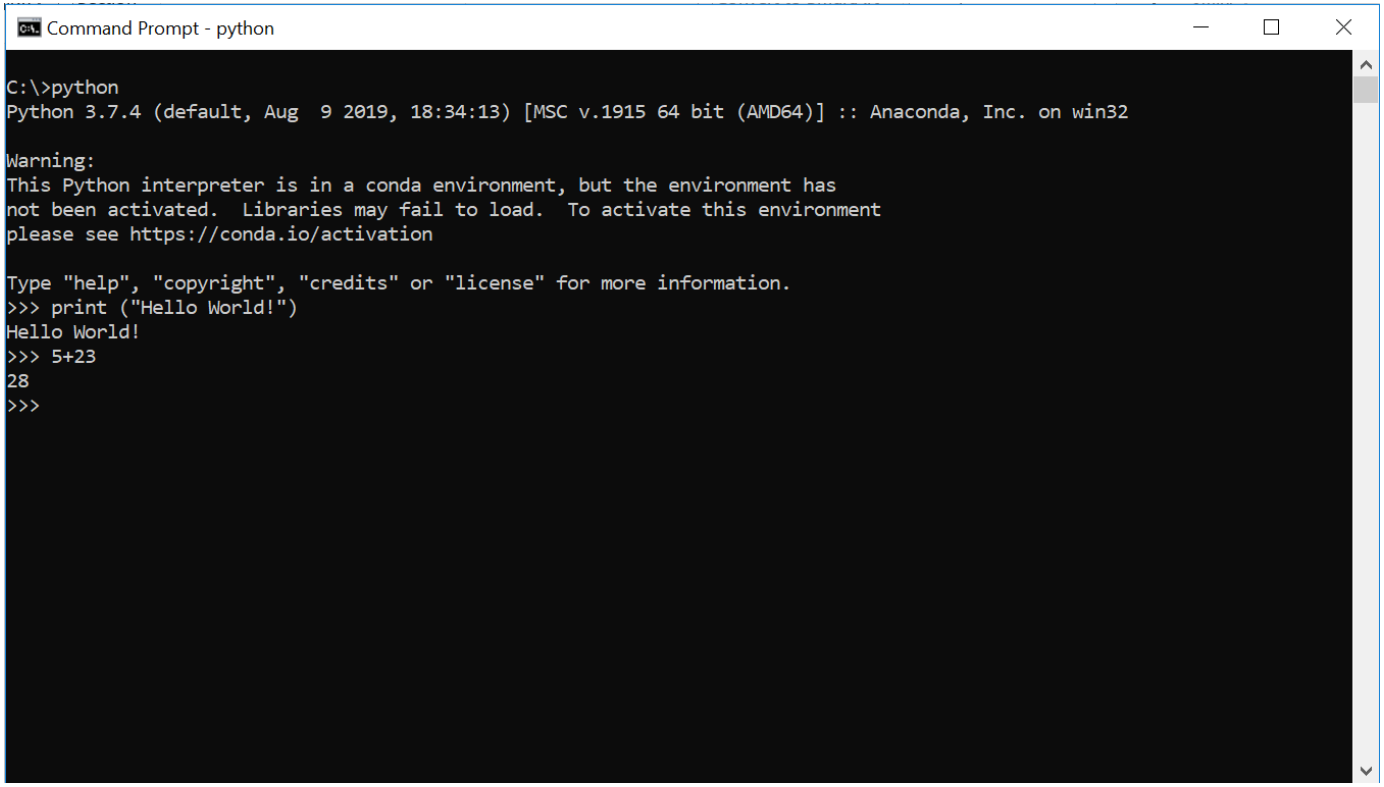
1. Python prompt displays (prints) your program output on the screen



Python Prompt via Command Prompt

Let's use Python Prompt just like a calculator

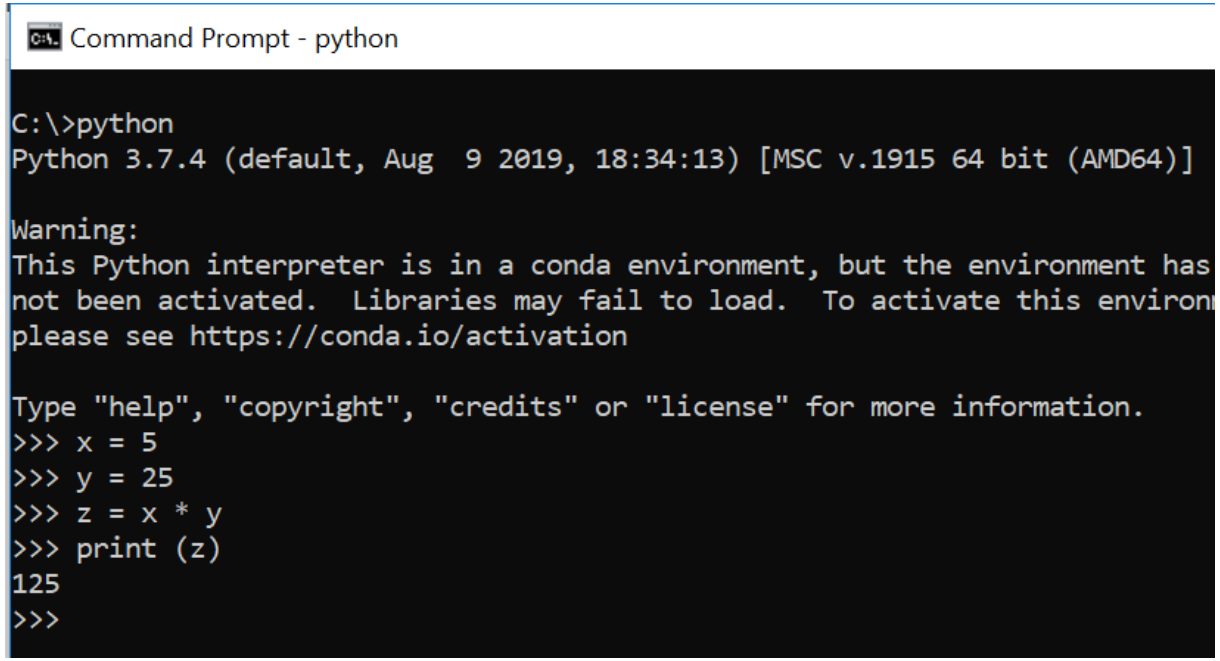
1. Enter 5 + 23 at the Python Prompt
2. Python Prompt automatically displays the results of 5+23



Python Prompt via Command Prompt

Let's write another program that will assign values to two variables then multiply these two variables to generate result and store it in a third variable

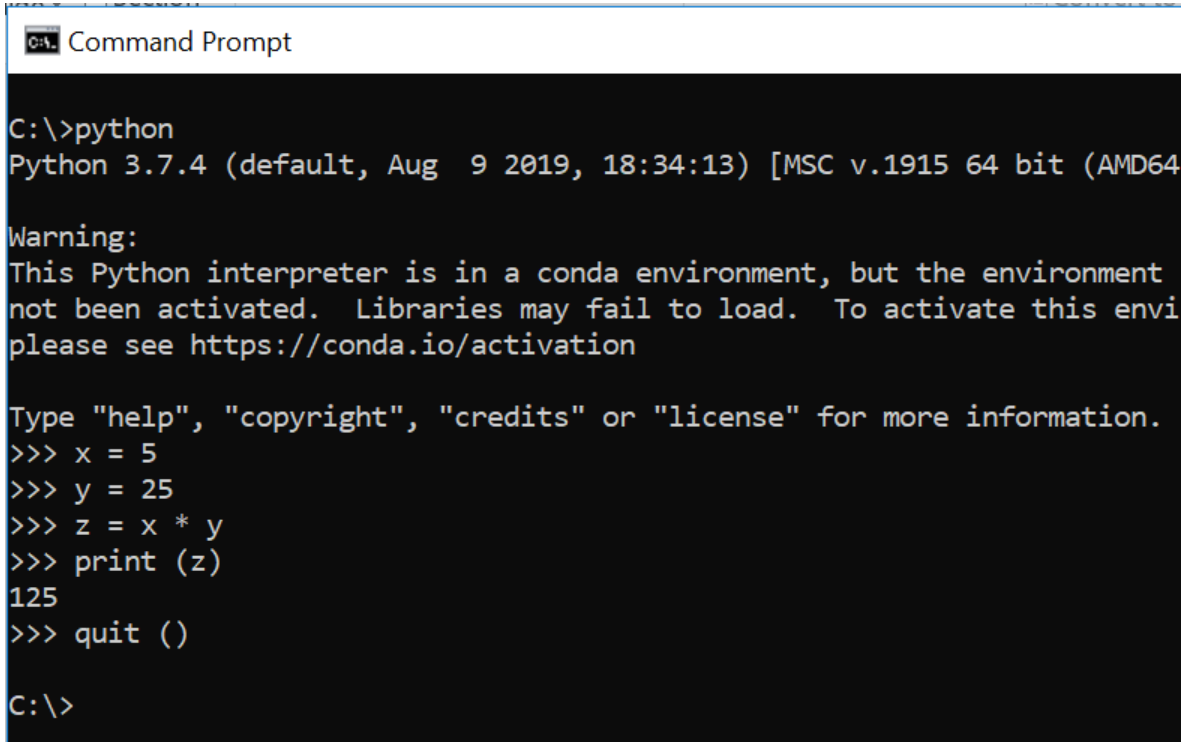
1. Enter x = 5 at the Python Prompt
2. Enter y = 25 at the Python Prompt
3. Enter z = x * y at the Python Prompt
4. Enter print (z) at the Python Prompt



Python Prompt via Command Prompt

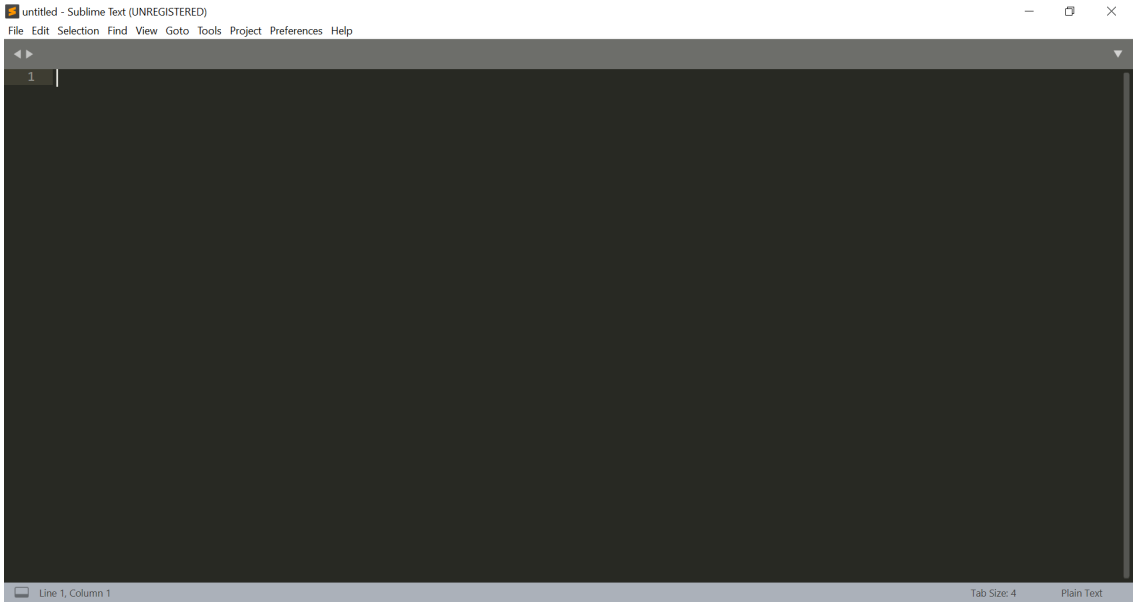
To end the Python Prompt session:

1. Enter quit()



Python Programming Using Editor

Python Programming Using Editor



Python Programming Using Editor

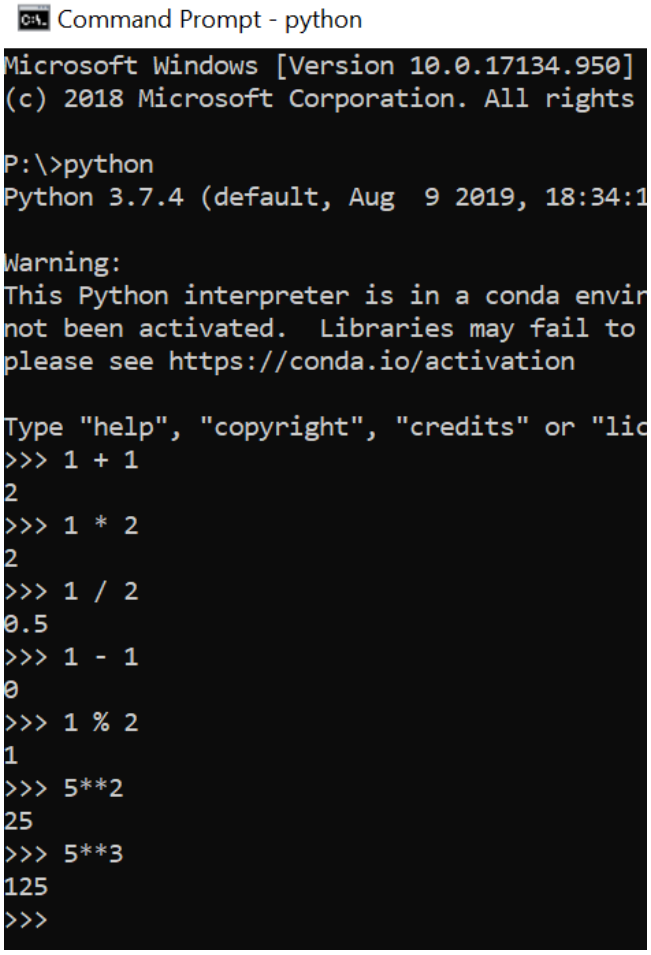
Data Science: Python Programming Arithmetic Operators

Data Science: Python Programming Arithmetic Operators

- + addition
- - subtraction
- * multiplication
- / division
- % mod
- ** to the power of

My Dear Aunt Sally applies!

Python Programming Arithmetic Operators



Data Science: Python Data Types

Data Science: Python Data Types

Integer numbers	int	100
Float, real numbers	float	100.0
Text, string	str	“data science” ‘data science’
Boolean	bool	True or False

Data Science: Python Data Types

Array		
Dictionary (unordered, changeable, indexed, no duplicates) (Can access items by [“key”])	dict	{“course": “Data Science Fundamentals", “mode": “Online", “term": “Fall 2019” }
Lists (ordered, changeable, duplicates are ok) (can access items by [index#])	list	[“Adam”, “Betsy”, “Cherry“, “Betsy”]
Tuples (ordered, unchangeable, duplicates ok) (can access items by [index#])	tuple	(“may”, “june”, “july”,”july”, August)
Sets (unordered, unindexed, no duplicates) (cannot access items by index#)	set	{"apple", "banana", "craneberry"}

Data Science: Python Data Types

Python: For Loop

For loop the traditional thinking….

For counter from 1 to 100
    Do…….

For Counter = 0, add 1 to the counter for each iteration until counter is less than X

Python: For Loop

a for loop iterates over a sequence

a list, a tuple, a dictionary, a set, or a string

For example:

```
genre =  ["Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
"Drama", "Family", "Fantasy", "History", "Horror", "Music", "Musical", "Mystery",
"Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western"]

count = 0
for value in genre:
   count = count + 1

print("There are ", count, " genres of movies in the list")
```

Python: For Loop

a for loop iterates over a sequence

a list, a tuple, a dictionary, a set, or a string

For example:

```
genre =  ["Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
"Drama", "Family", "Fantasy", "History", "Horror", "Music", "Musical", "Mystery",
"Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western"]

for value in genre:
    print(value)
```

Python: For Loop

a for loop iterates over a sequence

a list, a tuple, a dictionary, a set, or a string

For example:

```
genre =  ["Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
"Drama", "Family", "Fantasy", "History", "Horror", "Music", "Musical", "Mystery",
"Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western"]

for value in genre:
    print(value)
    if value == "Comedy“:
    break
```

Python: For Loop

a for loop iterates over a sequence

a list, a tuple, a dictionary, a set, or a string

For example:

```
for value in ‘data science’:
   print(value)
```

Python: For Loop

Range(num1:num2)

returns numbers from num1 to num2 minus 1 (excluding num2)

For example:

```
base = input("Enter an integer base number")
exponent = input("Enter a positive integer exponent number")
result = 1

base = int(base)
exponent = int(exponent)

for i in range(0, exponent):
   result = result * base

print(result)
```

Python While Loop

Loops as long as a test condition is true

Python: For Loop

For example:

```
base = input("Enter an integer base number")
exponent = input("Enter a positive exponent number")
result = 1

base = int(base)
exponent = int(exponent)

for i in range(0, exponent):
    result = result * base


print(result)
```

For example:

```
base = input("Enter an integer base number")
exponent = input("Enter a positive exponent number")
result = 1

base = int(base)
exponent = int(exponent)

i = 0
while i < exponent:
    result = result * base
    i = i + 1

print(result)
```

Python: For Loop

For example:

```
base = input("Enter an integer base number")
exponent = input ("Enter a positive integer exponent number")
result = 1

base = int(base)
exponent = int(exponent)

i = 0
while i < exponent:
    result = result * base
    i = i + 1

print(result)
```

Data Science: Python Built-in Functions

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Fruitful Function VS. Void Function (Procedure)

- Fruitful Function: returns a value to the caller
  - Int()
- Void Function (Procedure): perform tasks but not value returned to the caller
  - Print()

User-Defined Function (Fruitful Function)

- For example: we can turn our exponent code into a fruitful function:

```
def exponent(base, exp):
        result = 1

        for i in range(0, exp):
                result = result * base

        return result
```

Save this user-defined function as MyMathFunction.py in the same folder as the main program

```
import MyMathFunction

base = input("Enter an integer base number")
exp = input("Enter a positive integer exponent number")

base = int(base)
exp = int(exp)

eresult = MyMathFunction.exponent(base, exp)

print(base,"^",exp," = ",eresult)
```

This is the main program. Save this main program, in the same folder as MyMathFunfion.py

User-Defined Function (Void Function or Procedure)

```
def Compute_Mean(numbers=[0,0,0]):
    n = len(numbers)

    total = 0
    for value in numbers:
        total = total + value

    mean = total / n
    return (mean)
```

Save this user-defined function as StatMean.py in the same folder as the main program

```
from StatMean import Compute_Mean

Compute_Mean([1,1,1])
```

This is the main program.